CSC311 Project Report

Yanke Mao, Jiaxu Li, Yuran Zhang, Alex Denisov

December 2022

Abstract:

This report is separated into 5 sections; data, model, model choice, prediction and workload.

In the data section, we begin with performing an experimental data analysis (EDA) on the provided survey data. Also within this section, we include a number of graphs (boxplots and histograms) to represent the distributions of the values in the data set; using these graphs we will justify the features we selected and did not select to fit into a model.

In the model section, we will show the experiments run on a number of models and their performance results via sklearn. Using this initial performance data we will choose the most competitive model to use for the remainder of the project.

In the model choice section, we show the trained model and the hyperparameter tuning performed to optimize the model for our set metrics.

In the prediction section, we will state our initial performance prediction for our model, and supporting evidence for such a prediction.

The workload section will include a short paragraph from each group member describing their workload and their respective roles within the project.

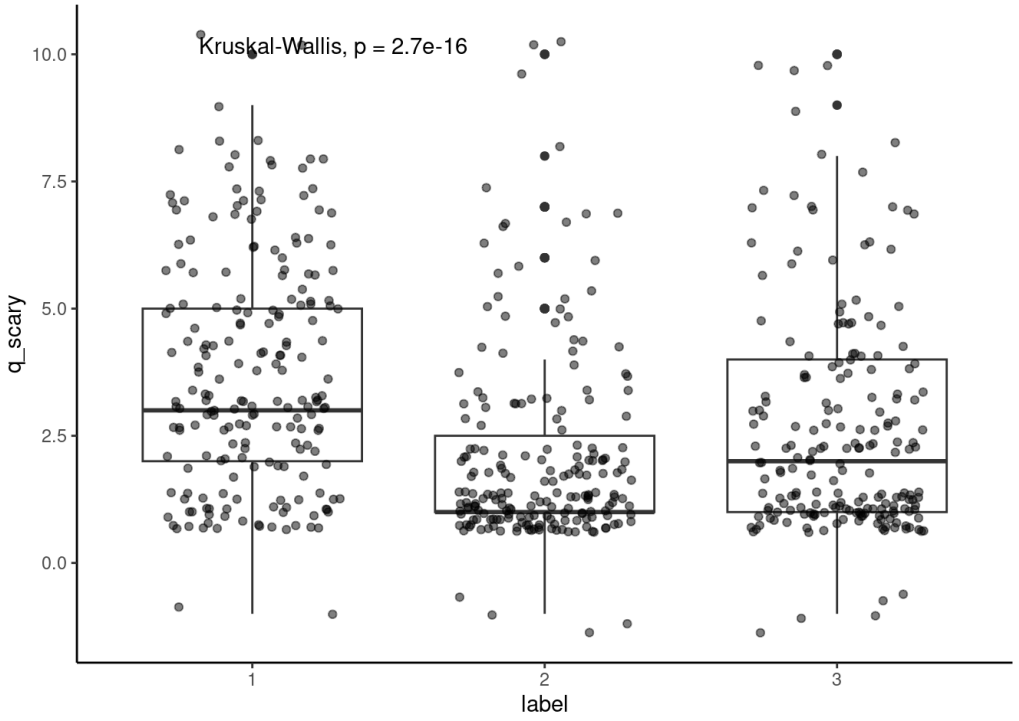
Data:

We have chosen to split the data into two separate data sets; a data set containing all of the stories for each image (q_story), and a data set that holds our selected remaining features as determined below in our EDA (q_remainder).

In further sections we will discuss the models in more detail; q_story is separated from the other features since it will be difficult to fit into a decision tree or logistic regression models, and will be better handled by naive bayes / gaussian discriminant analysis (GDA) or a k-NN model. The data in q_remainder is a better fit for use with models like logistic regression and CNN since it is easier to vectorize, whether discrete or continuous, and they are categorical. In further sections, we will discuss how we will fit these two data sets into two separate models and how they will be joined together to produce a single prediction.

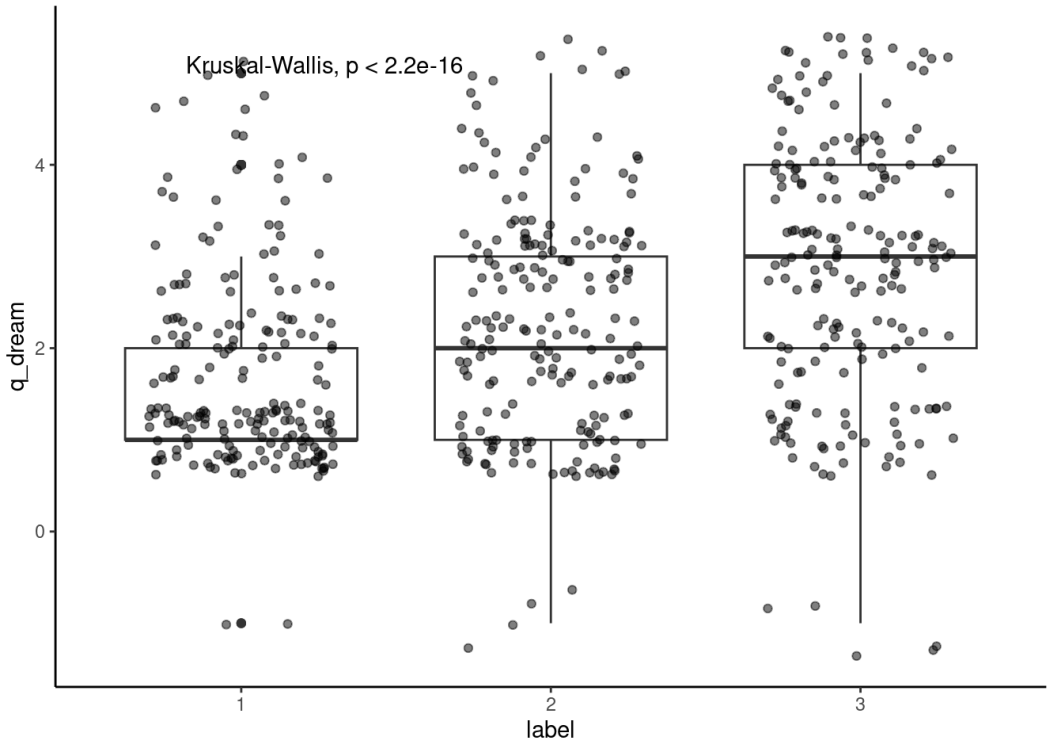
Below we use a combination of box plots and histograms. We use box plots for data where the average is sensitive to extreme data points found in the data set; we will use the median to better determine the correlation between the data and targets. The boxplots include a p-test value (Kruskal -Wallis test); this is a null hypothesis of whether the means are equal for the three classes. A small p-value ($p < 0.05$) indicates that the feature is distinguishable between the three pictures and a large p-value ($p > 0.05$) represents that we are unable to distinguish the classes from the averaged data sets. (See csc311_project.html)

Boxplot 1 - q_scary / image - chosen



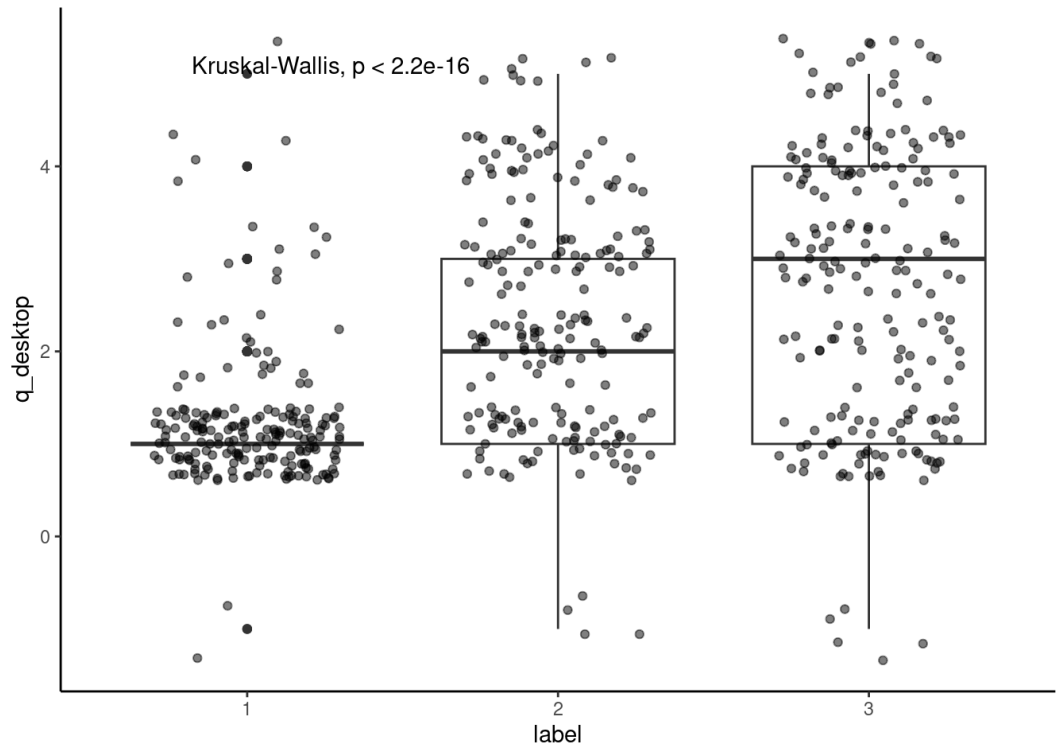
As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. This shows that we are able to distinguish data between labels (pictures) relatively well. As also seen in the graph the median values are separated and easily distinguishable from each other, thus we have chosen to use q_scary as a predictor for our model.

Boxplot 2 - q_dream / image - chosen



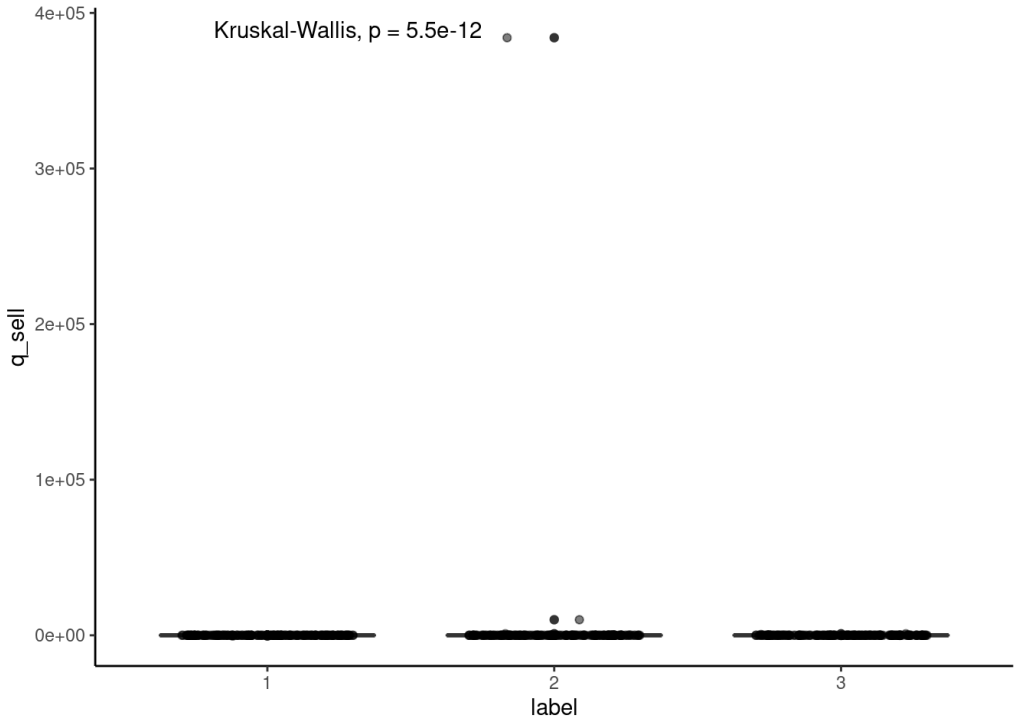
As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. This shows that we are able to distinguish data between labels (pictures) relatively well. As also seen in the graph the median values are separated and easily distinguishable from each other, thus we have chosen to use q_dream as a predictor for our model.

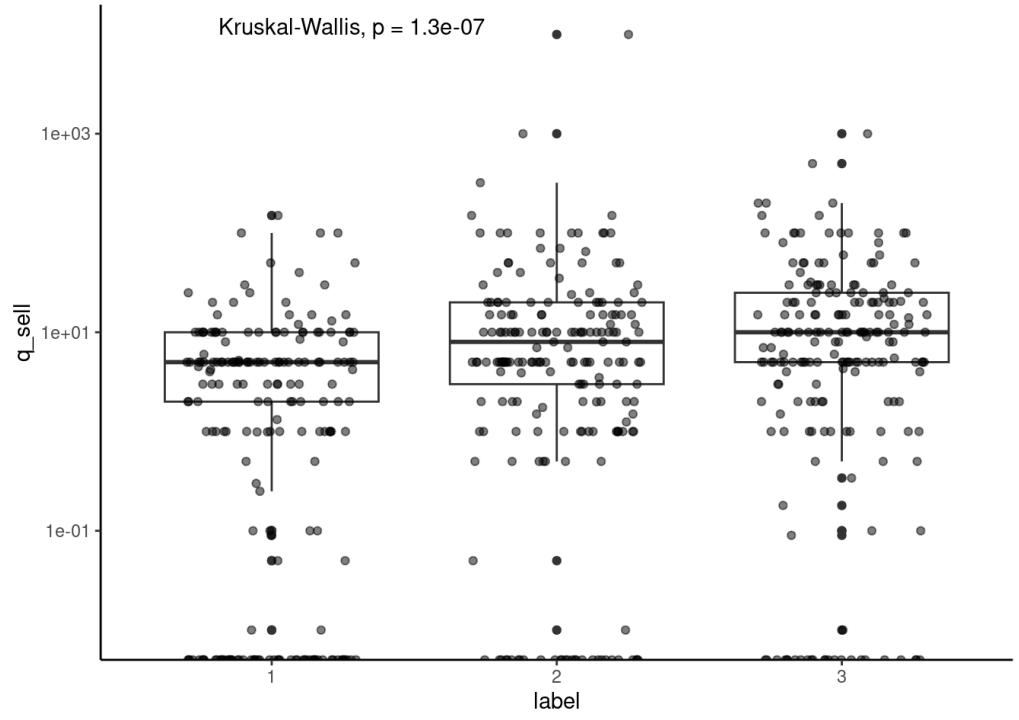
Boxplot 3 - q_desktop / image - chosen



As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. This shows that we are able to distinguish data between labels (pictures) relatively well. As also seen in the graph the median values are separated and easily distinguishable from each other, thus we have chosen to use q_desktop as a predictor for our model.

Boxplot 4/5 - q_sell / image & normalized q_sell / image - not chosen

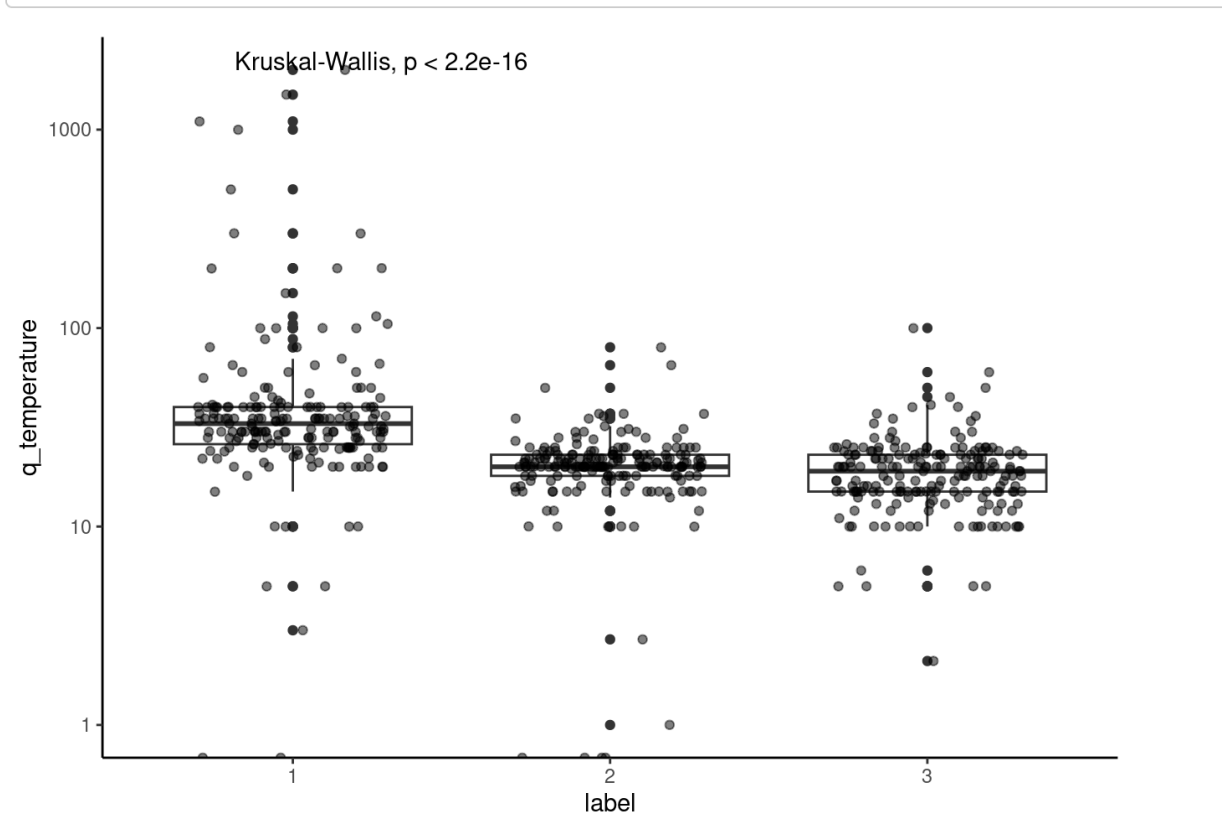




The first graph is the raw plot for q_sell; which includes a few outlying extreme values and thus makes the graph unreadable. The second graph is a plot of normalized q_sell data; where extreme values were removed and everything was multiplied by log10 to make it readable; while not significantly changing the meaning of the data for prediction.

As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. Except when viewed the medians are very close together, and all images have a large amount of 0 values which would make it hard to distinguish which data points belong where. While the p-test value is valid the graph shows that this is not an optimal choice to fit into our model, thus we choose not to use q_sell as a predictor for our model.

Boxplot 6 - normalized temperature/image - chosen

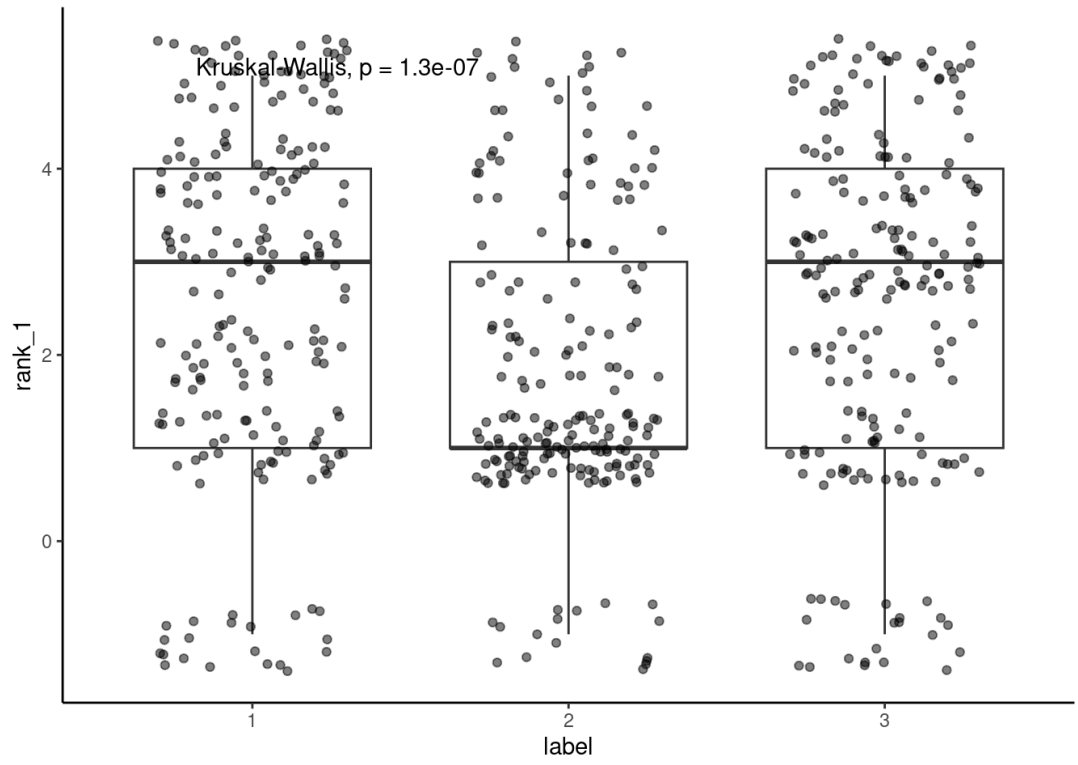


The above graph shows normalized temperature data; this was done for similar reasons to boxplot 4 / 5 removing extreme values and taking the log10 of all values.

As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. This shows that we are able to distinguish data between labels (pictures) relatively well. As also seen in the graph the median values are separated for image 1, and separated for image 2 and image 3; making image 1 easily distinguishable from the other images, thus we have chosen to use q_temperature as a predictor for our model as it will allow us to very clearly predict if something is image 1 or not.

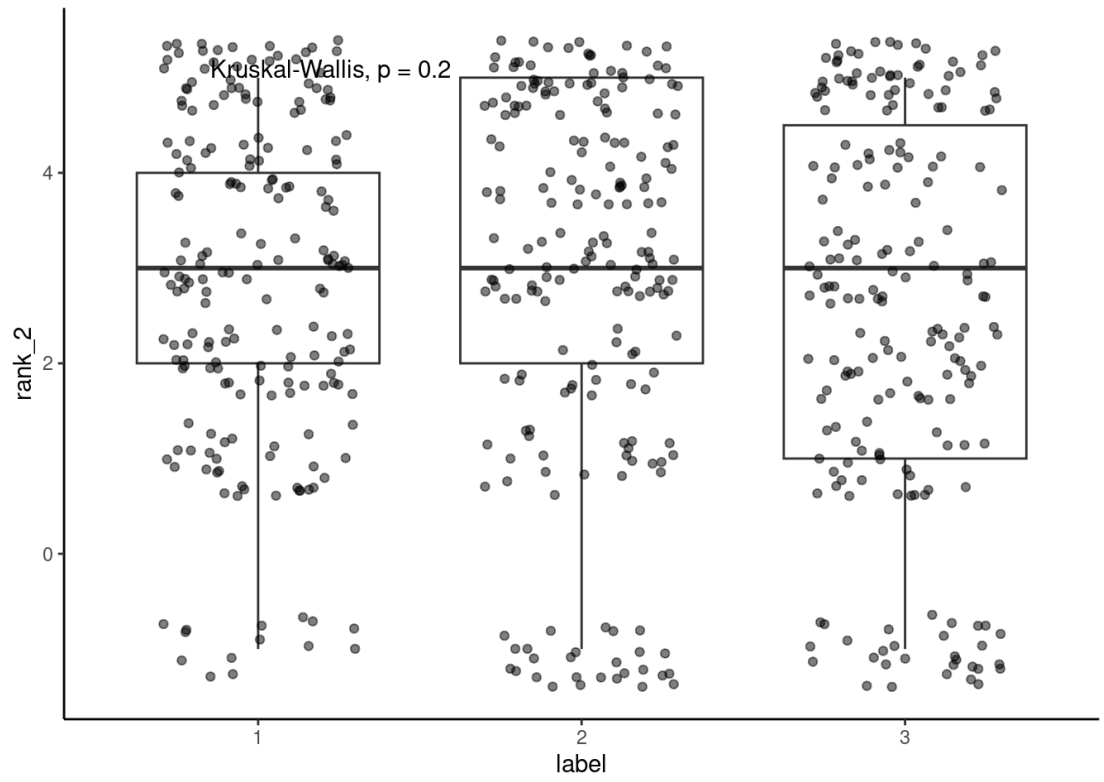
Below we have various boxplots for rank_#, we chose the split the q_quote data set into five parts, where the number is in respect to its position in the list of quotes. This is done to better determine which quotes correlate with which images.

Boxplot 7 - rank_1 / image - chosen



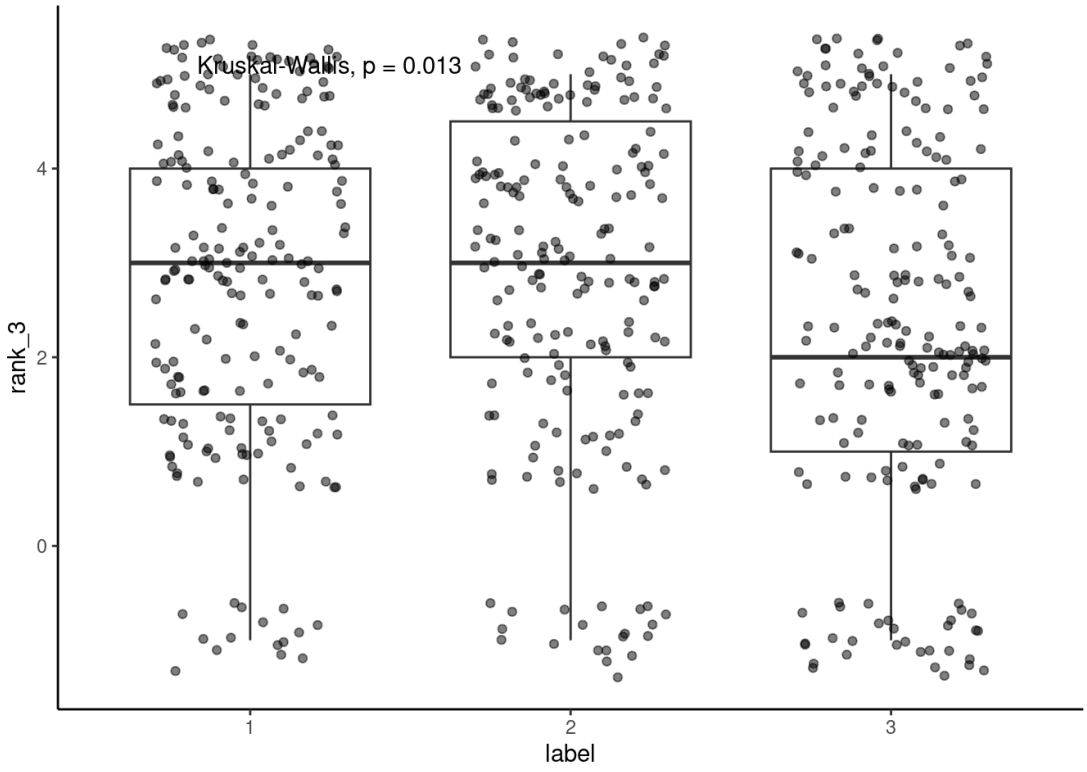
As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. This shows that we are able to distinguish data between labels (pictures) relatively well. As also seen in the graph the median values are separated for image 2, and separated for image 1 and image 3; making image 2 easily distinguishable from the other images, thus we have chosen to use rank_1 as a predictor for our model as it will allow us to very clearly predict if something is image 2 or not.

Boxplot 8- rank_2 / image - not chosen



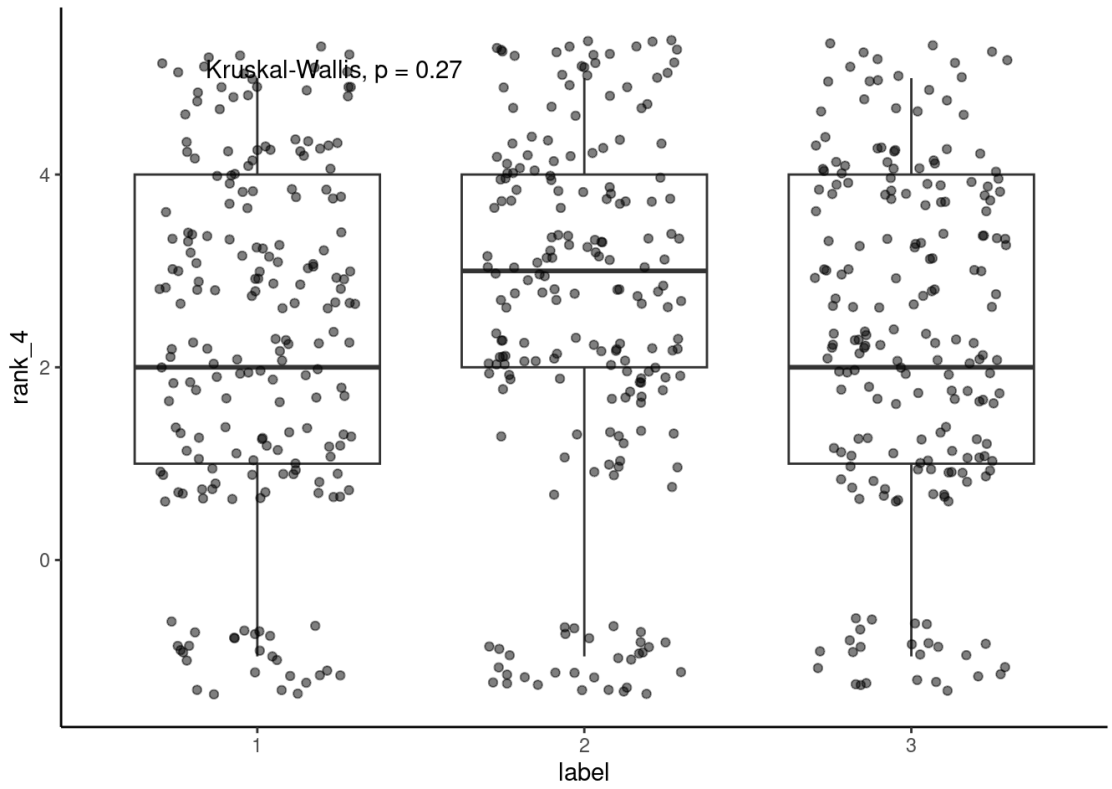
As seen in the above boxplot our p-test value is very high (0.2); we aim for p-test values smaller than 0.05 for the purposes of this project. This means that we are unable to distinguish between our labels (images) for rank_2. As seen in the graph the median values are all very close together and thus the data is non-distinguishable, hence we do not select to use rank_2 as a predictor in our model.

Boxplot 9- rank_3 / image - chosen



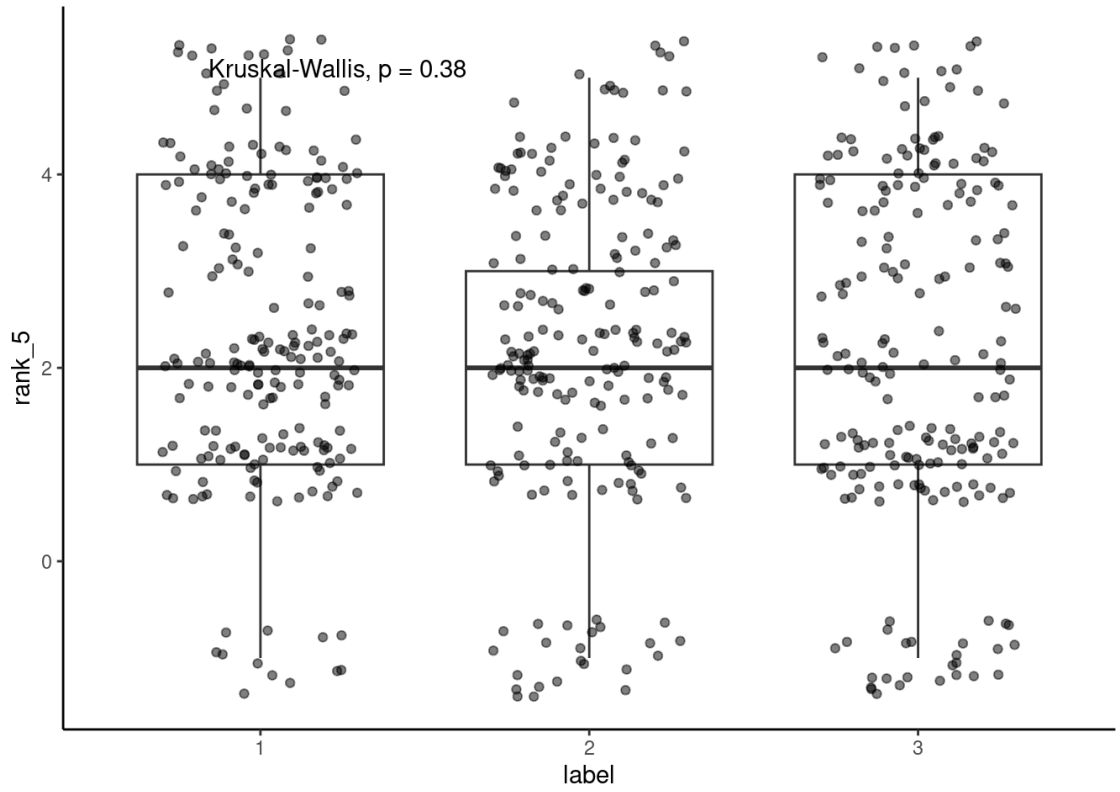
As seen in the above boxplot our p-test value is very small; significantly lower than our minimum p-test value for this project of 0.05. This shows that we are able to distinguish data between labels (pictures) relatively well. As also seen in the graph the median values are separated for image 3, and separated for image 1 and image 2; making image 3 easily distinguishable from the other images, thus we have chosen to use rank_3 as a predictor for our model as it will allow us to very clearly predict if something is image 3 or not.

Boxplot 10- rank_4 / image - not chosen



As seen in the above boxplot our p-test value is very high (0.27); we aim for p-test values smaller than 0.05 for the purposes of this project. This means that we are unable to distinguish between our labels (images) for rank_4. As seen in the graph the median values for image 1 and image 3 are quite close together, but image 2 is not. Since we already have a feature that is able to distinguish very clearly from image 2 and not image 2 (rank_1 / boxplot 7) we do not choose rank_4 as a predictor in our model.

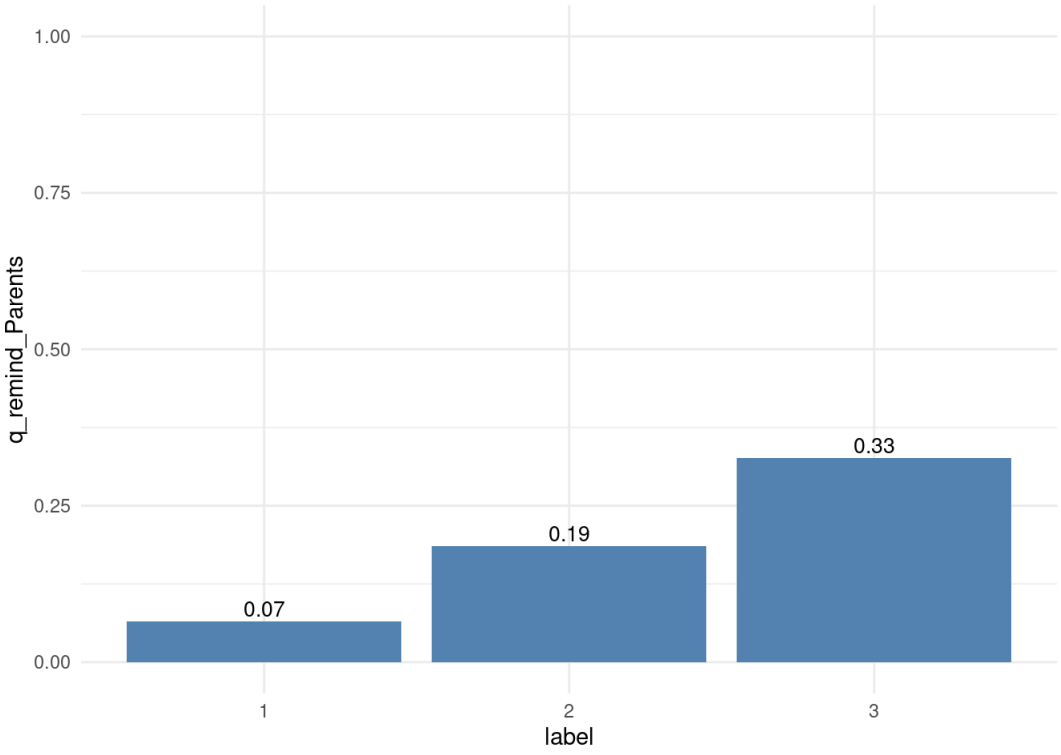
Boxplot 11- rank_5 / image - not chosen



As seen in the above boxplot our p-test value is very high (0.38); we aim for p-test values smaller than 0.05 for the purposes of this project. This means that we are unable to distinguish between our labels (images) for rank_5. As seen in the graph the median values are all very close together and thus the data is non distinguishable, hence we do not select to use rank_5 as a predictor in our model.

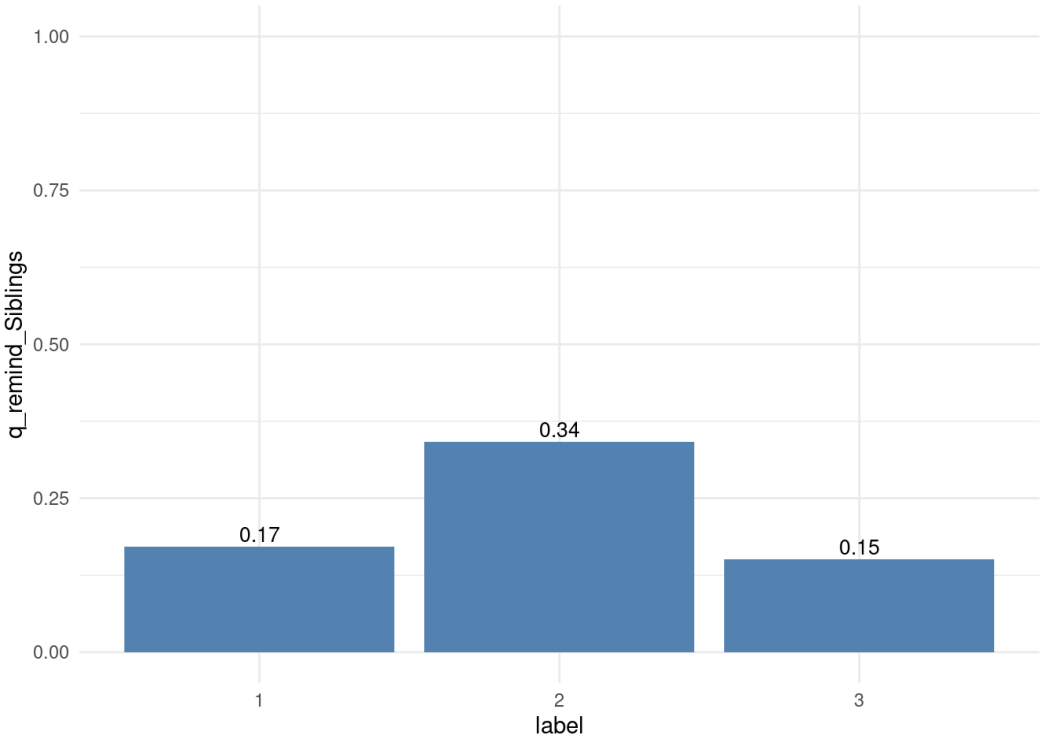
Below we have various histograms for q_remind_X, we chose to split the q_reminds data set into four parts, where X represents the selected item that the picture reminded the answerer of. This is done to better determine which items in the reminded list correlate with which images.

Histogram 1 - q_remind_parents / image - chosen



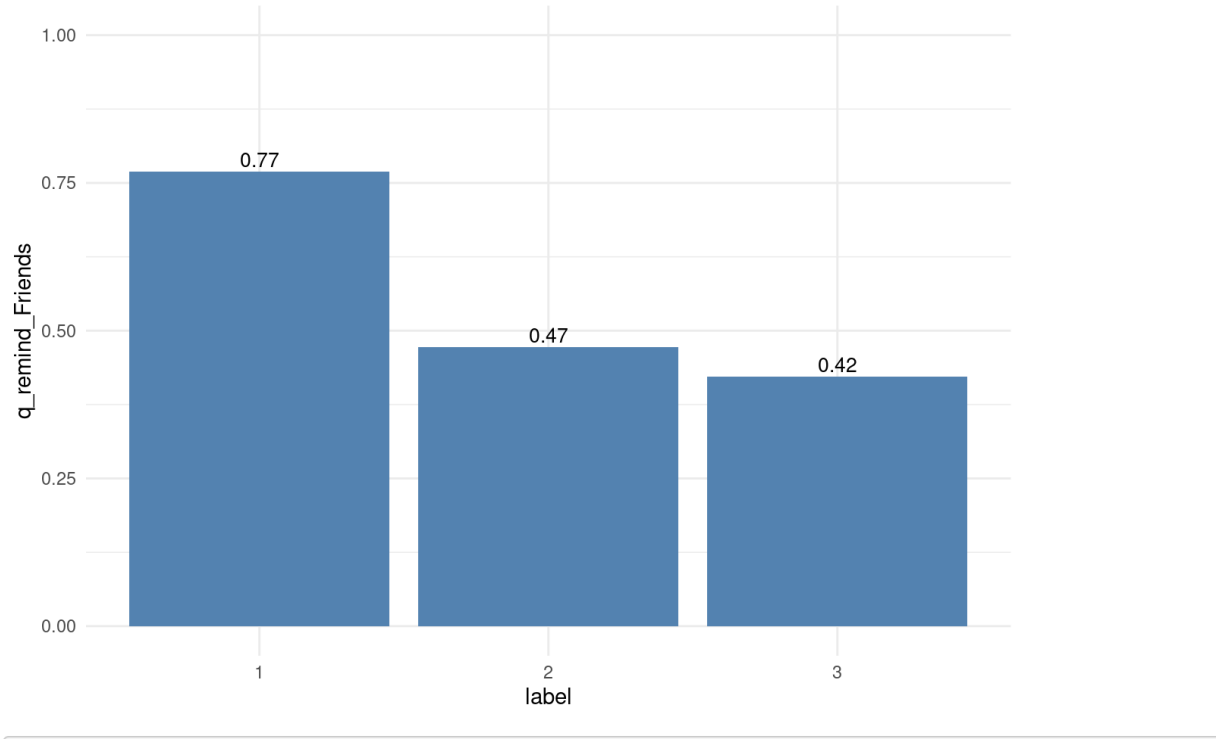
As seen in the above histogram we have a relatively uneven distribution of values for each image; making them distinguishable from each other. Thus we will use q_remind_parents as a predictor in our model as it is able to distinguish between all three images relatively well.

Histogram 2 - q_remind_siblings / image - chosen



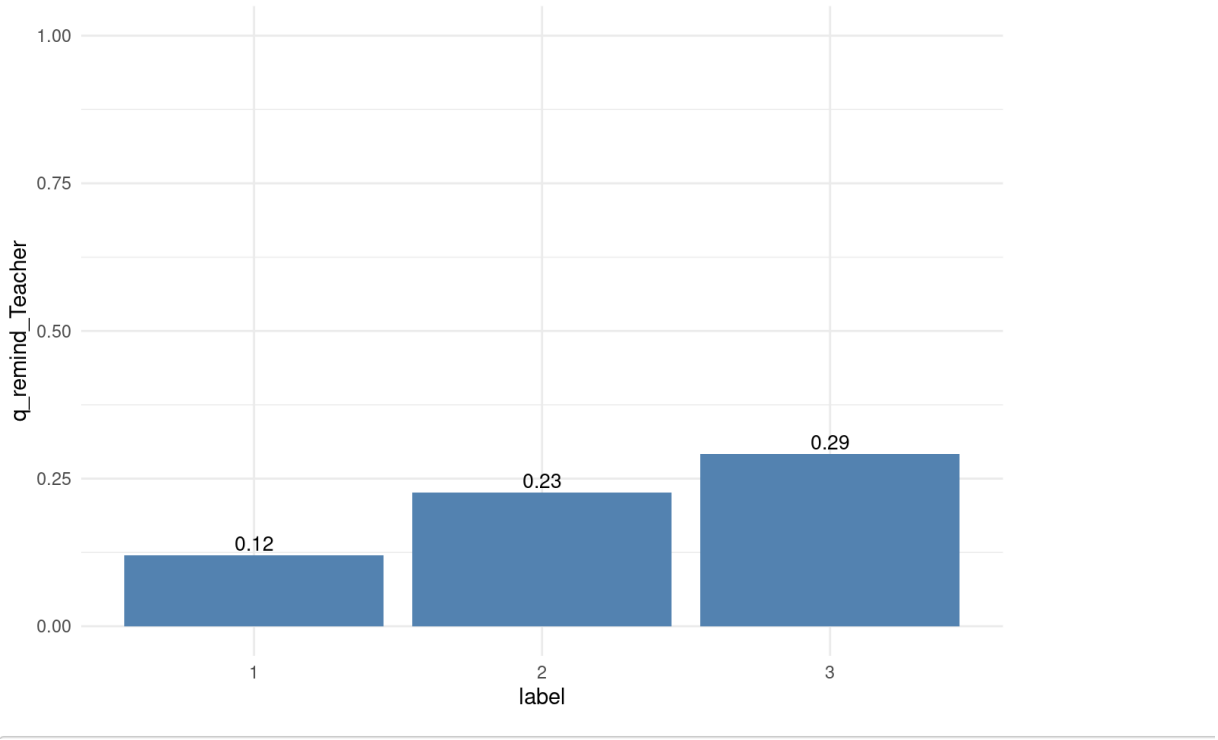
As seen in the above histogram we have a relatively even distribution of values for two images; making two hard to distinguish from each other but making image 2 distinguishable from the others Thus we will use q_remind_siblings as a predictor in our model as it is able to distinguish where the image is image 2 or not relatively well.

Histogram 3 - q_remind_friends / image - chosen



seen in the above histogram we have a relatively even distribution of values for two images; making two hard to distinguish from each other but making image 1 distinguishable from the others Thus we will use q_remind_friends as a predictor in our model as it is able to distinguish when the image is image 2 or not relatively well.

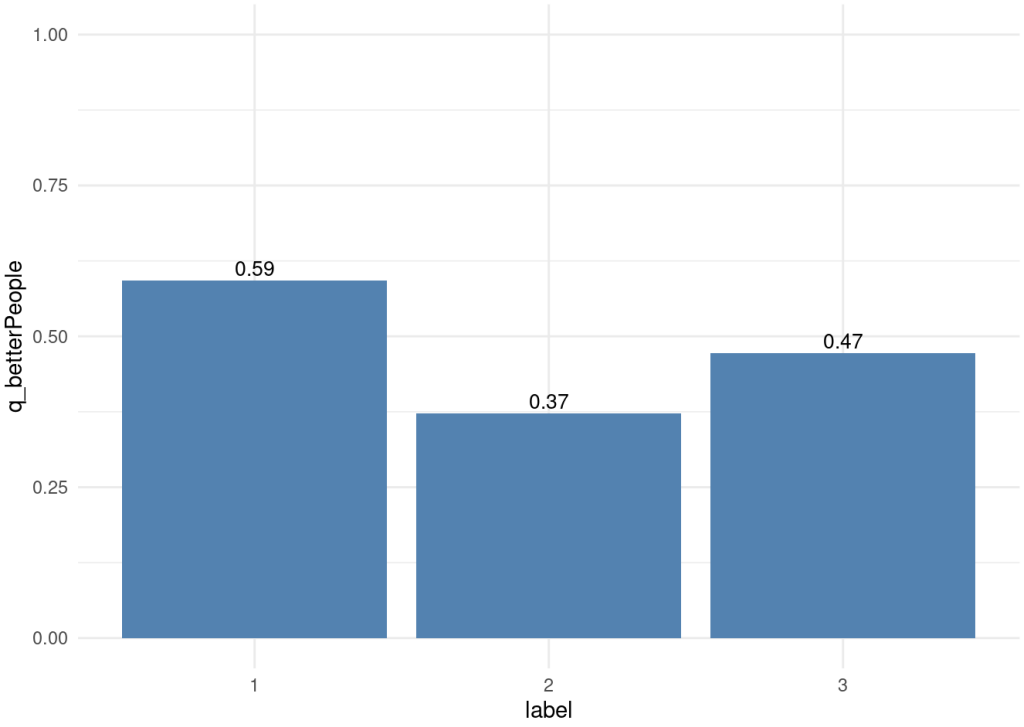
Histogram 4 - q_remind_teacher / image - chosen



As seen in the above histogram we have a relatively uneven distribution of values for each image; making them distinguishable from each other. Thus we will use q_remind_teacher as a predictor in our model as it is able to distinguish between all three images relatively well.

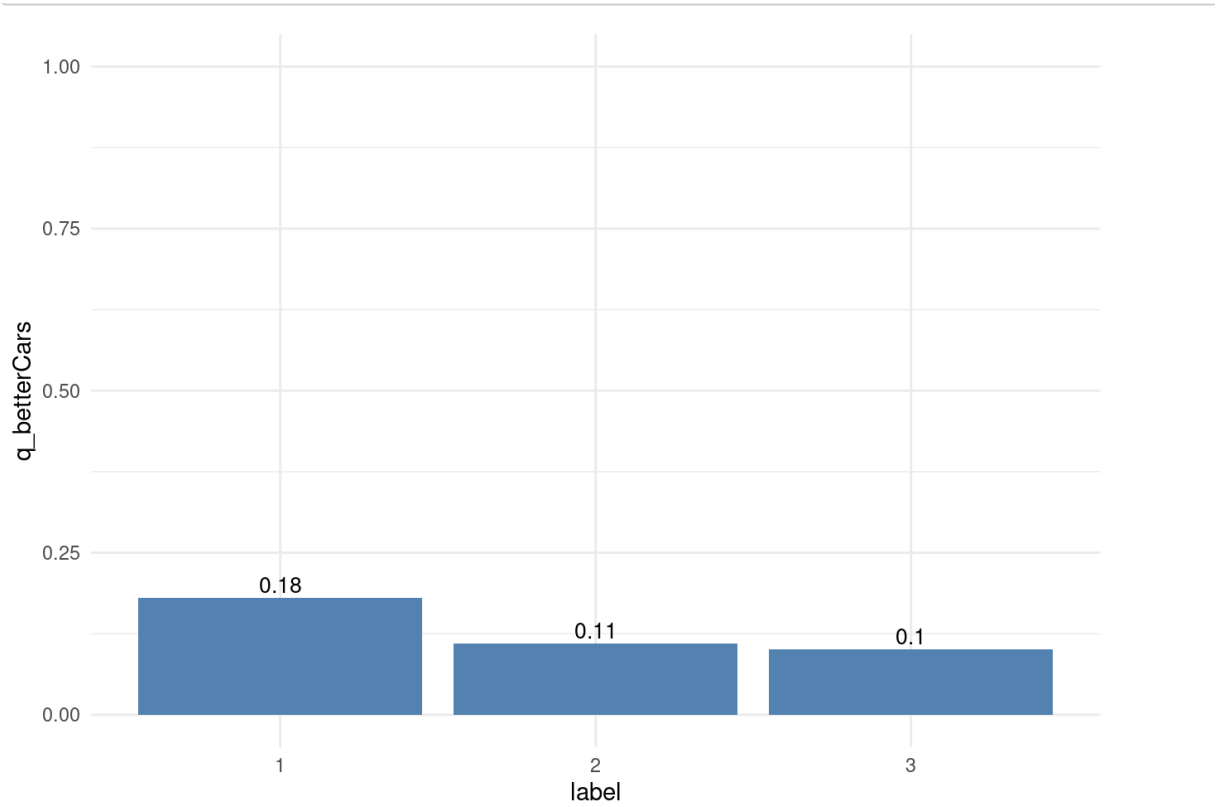
Below we have various histograms for q_better_X, we chose to split the q_better data set into five parts, where X represents the selected item that would make the picture better by the answerer. This is done to better determine which items in the better list correlate with which images. Histograms are used for these data points as they tolerate duplicate / multiple answers.

Histogram 5 - q_better_people / image - chosen



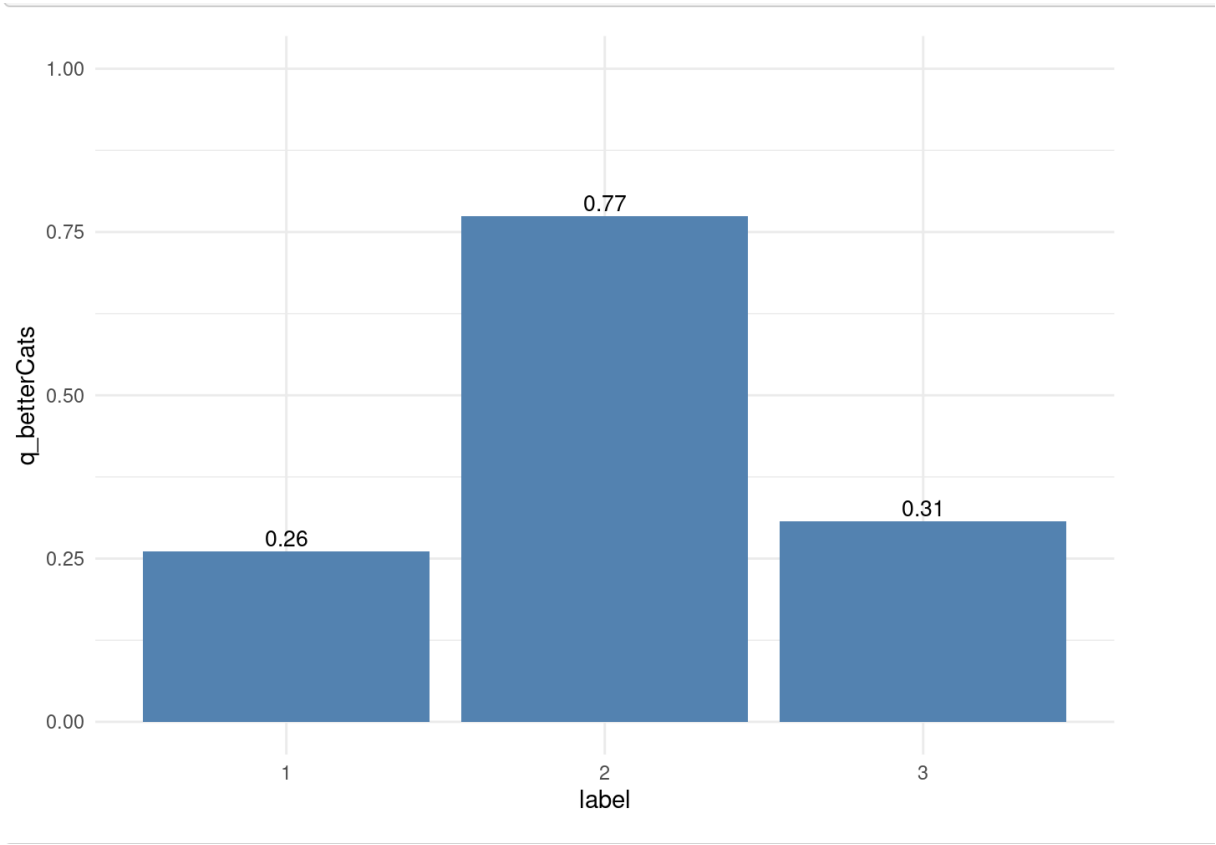
As seen in the above histogram we have a relatively uneven distribution of values for each image; making them distinguishable from each other. Thus we will use q_better_people as a predictor in our model as it is able to distinguish between all three images relatively well.

Histogram 6 - q_better_cars / image - chosen



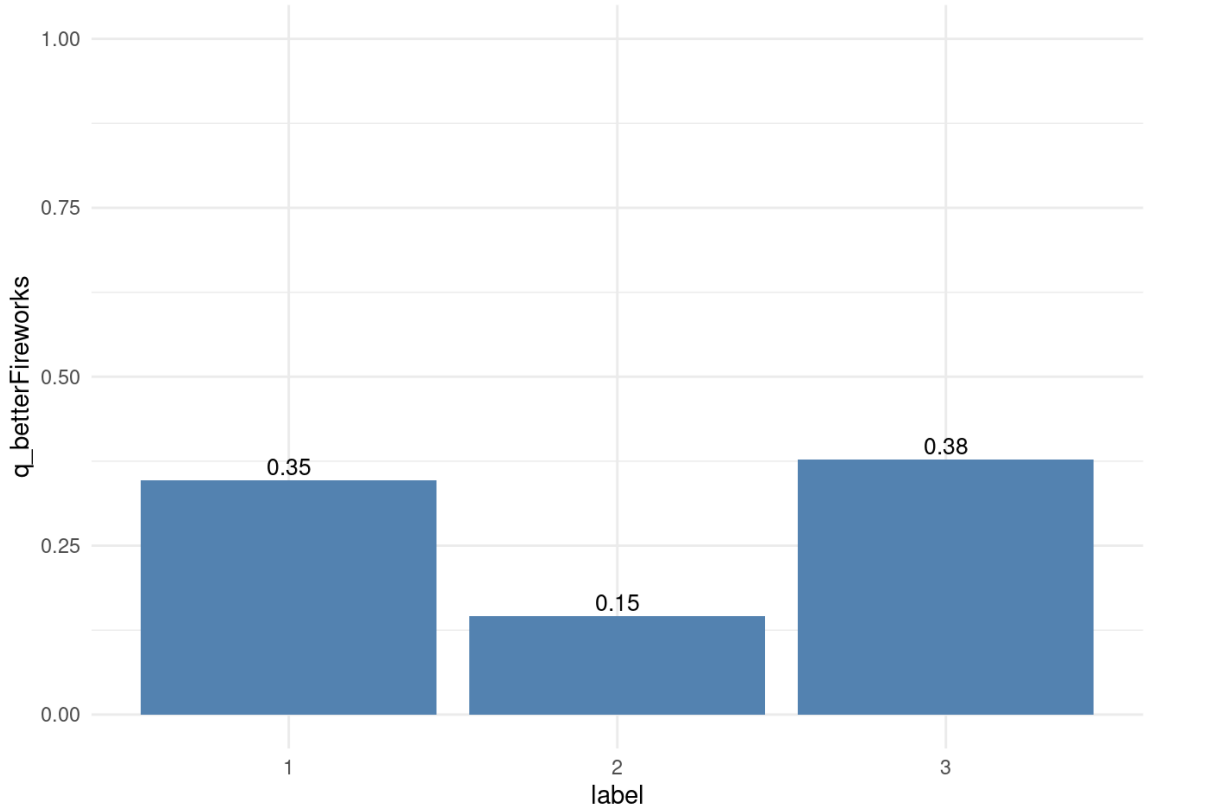
As seen in the above histogram we have a relatively even distribution of values for two images; making two hard to distinguish from each other but making image 1 distinguishable from the others Thus we will use q_better_cars as a predictor in our model as it is able to distinguish where the image is image 1 or not relatively well.

Histogram 7 - q_better_cats / image - chosen



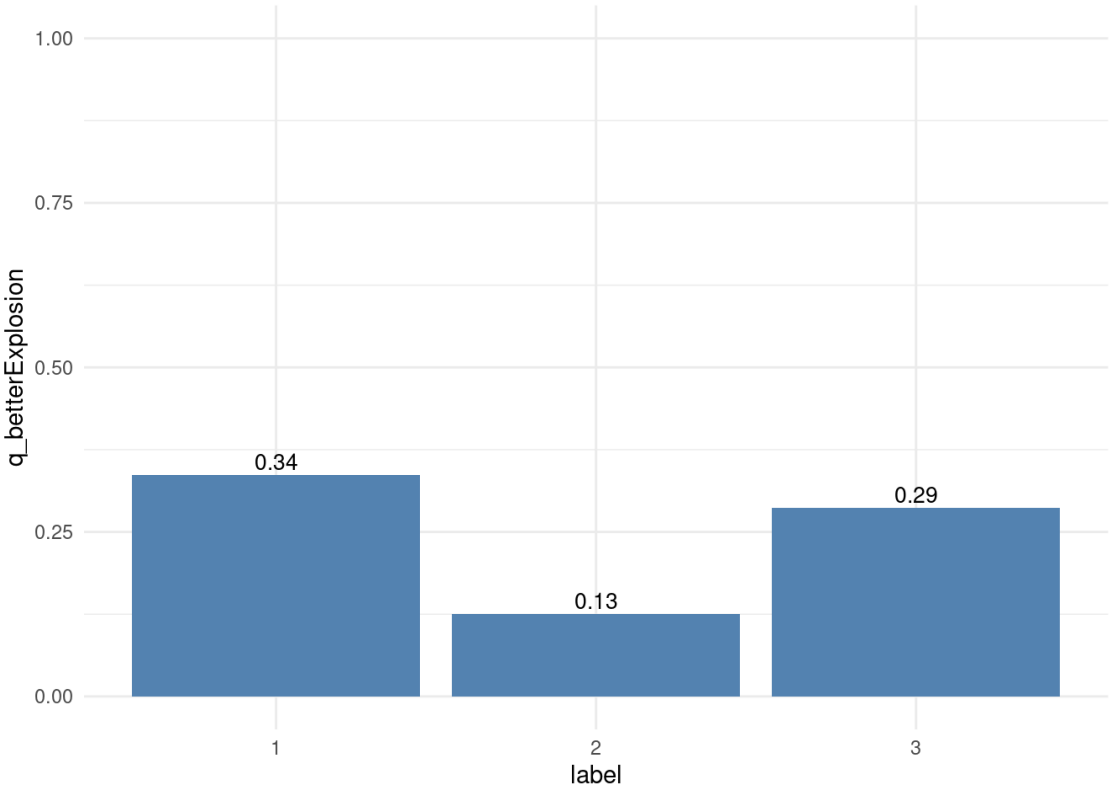
As seen in the above histogram we have a relatively even distribution of values for two images; making two hard to distinguish from each other but making image 2 distinguishable from the others Thus we will use q_better_cats as a predictor in our model as it is able to distinguish where the image is image 2 or not relatively well.

Histogram 8 - q_better_fireworks / image - chosen



As seen in the above histogram we have a relatively even distribution of values for two images; making two hard to distinguish from each other but making image 2 distinguishable from the others Thus we will use q_better_fireworks as a predictor in our model as it is able to distinguish where the image is image 2 or not relatively well.

Histogram 9 - q_better_explosion / image - chosen



As seen in the above histogram we have a relatively even distribution of values for two images; making image 1 and image 3 hard to distinguish from each other but making image 2 distinguishable from the others Thus we will use q_better_explosions as a predictor in our model as it is able to distinguish where the image is image 2 or not relatively well.

In the above data, q_story is not mentioned; the data in q_story cannot easily be represented in a box plot or histogram as easily, since there are ~4000 unique words in the data set. This has caused us to separate q_story into its own model. We are doing this due to previous course experience that shows that a large data set of words is significant enough to predict the positivity of reviews (classify bags of words into classes), thus it is likely that the bag of words is likely a good predictor for image labels.

Thus we will separate the data into two models:
q_remainder will be defined as the features from the data we have chosen to use in model 1; they are 'q_scary', 'q_dream', 'q_desktop', 'rank_1', 'rank_3', 'q_remind', 'q_better', 'q_temperature'.

Model 1 will use q_remainder for the reasons shown above in each data set's respective table and paragraph. ‘Q_scary’, ‘q_dream’ and ‘q_desktop’, ‘q_remind’ will be one-hot encoded in the model. q_quote is represented as a one hot 5x5 matrix; extracting rank_1 rank_3 we get two one hot vectors with a length of 5. q_temperature is stored as an indicator; if the value is greater than the median then the result is 1, 0 otherwise.

Model 2 will use q_story separated into a vocabulary to create a bag of words representation for each user answer.

Model:

We evaluated four models for model 1 and compared their performances on our chosen data set; we completed a k-NN model, a logistic regression model, a decision tree model and a Naive-Bayes model. All four of these models utilize q_remainder as defined above. We picked the below four models since we are dealing with a classification problem, and these are the previously known models that are able to handle classification well. We compare these four models together using their test accuracies and then pick the classification model with the highest accuracy.

k-NN: For our first trial we extended the provided sklearn k-NN model that was given in the starter code, we do not use q_story as it will not be processed by model 1. We found that the recommended k value for a k-NN model is roughly equivalent to the square root of n; since we have around ~600 data points. The optimal value for k should be around 20 - 25. For our k-NN model, we found that the optimal k value was 23 and achieved the following results(Knn.py):
KNeighborsClassifier training accuracy: 0.554
KNeighborsClassifier test accuracy: 0.5360824742268041

Logistic Regression: For our second trial we used the sklearn logistic regression model to test its possible accuracy for further implementation. For logistic regression, we used a subset of q_remainder, which is: 'q_scary', 'q_desktop', 'rank_2', 'rank_3', 'q_better', without q_temperature. We achieved the following results(log_testing.py):
Logistic Regression test accuracy: 0.7422680412371134

Decision Tree: For our third trial we used the sklearn decision tree model to test the possible accuracy of the model for further development. For the decision tree model, we used our previously defined q_remainder data set and achieved the following results(desisionTree.py):
DecisionTreeClassifier training accuracy: 0.984
DecisionTreeClassifier test accuracy: 0.5979381443298969
This is very likely an over-fit. Since we have too many features and too little data; we likely have only a single feature per leaf in the decision tree, and would perform unexpectedly on previously unseen data.

Naive-Bayes: For our fourth trial we tested the sklearn Bernoulli Naive-Bayes model to test its possible accuracy for further implementation with our predefined q_remainded data set to achieve the following results(Bayes.py):
BernoulliNB training accuracy: 0.736
BernoulliNB testing accuracy: 0.7319587628865979

As seen in the results above Naive-Bayes and logistic regression have very similar accuracy and performance. For simplicity, we will choose the Naive-Bayes model for model 1 as it is one of the two models with the highest accuracy of all the models.

We evaluated two models for model 2 and compared their performances on our chosen data set; we completed a k-NN model and a Naive-Bayes model. Both of these models utilize q_story as defined above. We chose to use the k-NN and Naive-Bayes model since our q_story data set is made up of ~4000 unique words; In this case, two models have both been shown to be effective for datasets like this.

k-NN: For our first trial we used the sklearn k-NN model to test its possible accuracy for further implementation. We used the bag of words to represent the vocabulary from stories represented as q_story. By iterating value k from 1 to 31, k=15 gives the highest accuracy. Thus we achieved the following results:
KNeighborsClassifier test accuracy: 0.4742268041237113

Naive-Bayes: For our second trial we used the sklearn Naive-Bayes model to test its possible accuracy for further implementation. We used the bag of words to represent the vocabulary from stories represented as q_story. We achieved the following results(BerulliBayesStory.py):
BernoulliNB test acc: 0.711340206185567
As seen in the results above we will choose the Naive-Bayes model for model 2 as it has the highest testing accuracy of the two models.

Model Choice and Hyperparameters:

As mentioned above we have chosen two models, model 1 for q_remainder using Naive-Bayes and model 2 for q_story also using Naive-Bayes. As previously mentioned q_remainder is a subset of our total data which has: 'q_scary', 'q_dream', 'q_desktop', 'rank_1', 'rank_3', 'q_remind', 'q_better', 'q_temperature' all being represented as one-hot vectors. With q_story being a bag of words representation of our story data.

In order to generate representative training, validation, and testing set. We intend to perform a random split to ensure the data with different labels are evenly distributed across all groups. But before the random split, we have to split the data by "user_id" to exclude the contamination of the test set. This step is crucial because if we perform a random split directly on the dataset. The prediction of the same people could be split into a training set and a test set respectively. However, this contaminates the test set since it will make the data in the training set and test set dependent. Classifying data by user_id before the random sample enables the data to be independent from each other.(See detail in Bayes.py)

By using the Python Panda library's sample function, we are able to create a reasonable split. The reason why we choose random split is that in models like Naive Bayes is that the naive assumption ensures that given the label, the data is independent of each other. Hence the split of the data will not affect the performance of the model to a great extent. Also, the size of the dataset we have is relatively small. Performing other splitting strategies like k-fold cross-validation is redundant in this case.

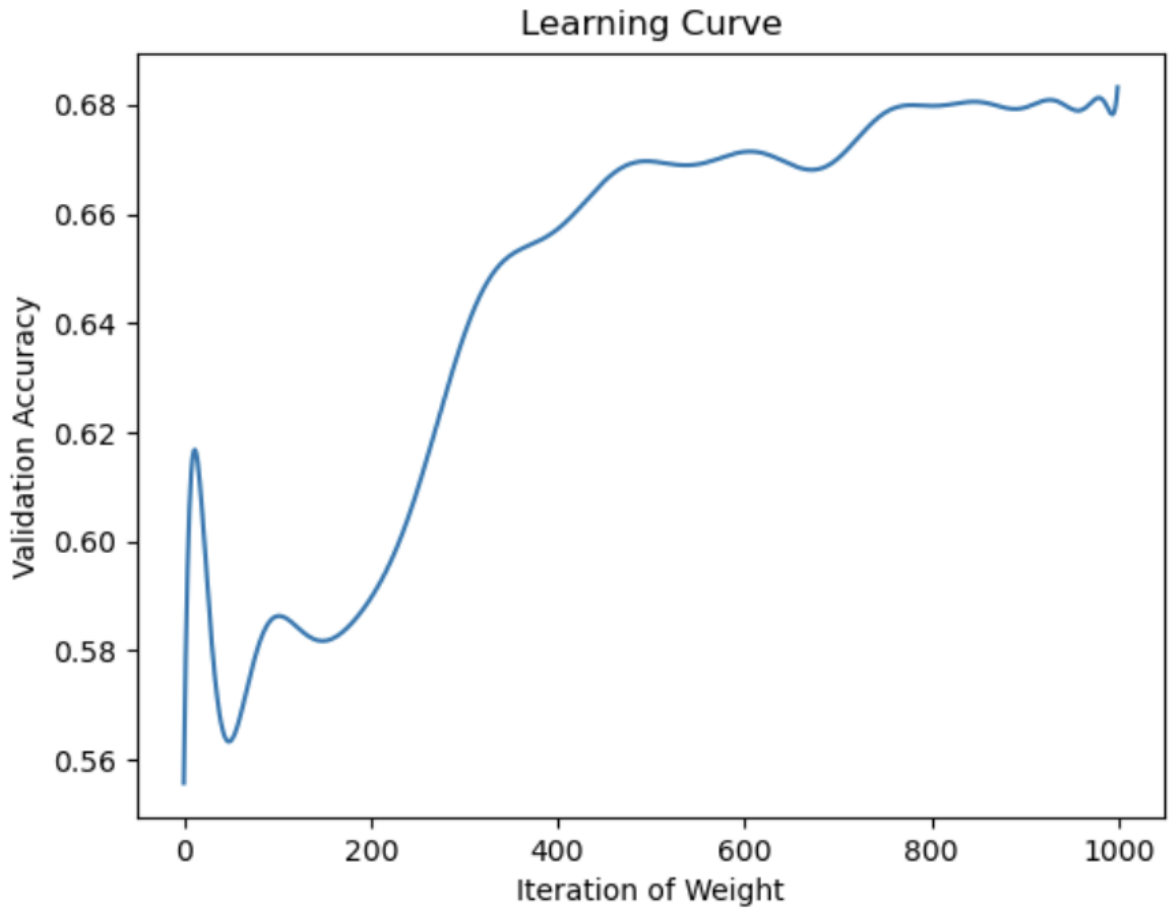
To evaluate our model we will look at the validation and testing accuracy results of the model. We initially thought to use the f1 score, but it was found that it was not optimal as it is only used for binomial classification problems; it is thus not applicable to our multi-feature classification problem. The f1 score is the harmonic mean of the precision and the recall; these are composed of the values from true positive, false positive and false negative. It is possible to calculate the f1 score for each label individual but it is impossible to do so for multiple features at once. Hence we would simply use classification accuracy as our evaluation metric instead.

If we were to generate a random number between 1 to 3 as the prediction of the data label, we would be able to reach at least a 33 percent accuracy. Thus we aim to double the test accuracy; which would be around 65 percent. However, we understand that the testing accuracy may be biased and only

represent limited information. We combine it with the training and validation accuracy to check the overfit and underfit of the model. We believe with a detailed EDA and the characteristics of s Naive-Bayes model will give us enough information to determine the quality of the model.

By the design of our model, since both the Naive-Bayes models output the probabilities; which are combined and the label with the highest probability is selected. In addition, model 1 and model 2 are independent of each other and look at different features. The Naive-Bayes model will use Maximum-A Posteriori (MAP) probability; the two hyperparameters for a Naive-Bayes model we have to tune are α and β in beta distribution, for this project, we will define α_1 and β_1 as being parameters for model 1, with α_2 and β_2 being the parameters of model 2. For both models, α_X and β_X are the hyper-parameter that we must use to tune our model; we pre-set the values of both α and β to α and β of the uniform beta distribution since we have no prior distribution and thus no prior presets for each.

Once we combine model 1 and model 2; we will call this combined model, model 3. Model 3 will have two hyperparameters to tune, let them be defined as γ_1 and γ_2 (where $\gamma_2 = 1 - \gamma_1$). γ_X represents the respective weight applied to each model's prediction. Model 3 combines the prediction of model 1 and model 2 with a weight applied to each; the predicted image is selected by taking the maximum combined prediction likelihood of the two models. We tune these γ_X hyperparameters based on the validation accuracy of the model. We found that the accuracy converges when $\gamma_1 = 0.733$. We hence set the weight to be 0.733; this is also because we want to bias against model 2. Model 2 is tuned by γ_2 and setting it to a lower value will make it less valuable for the overall prediction of the label, since model_2 is more likely to overfit; as mentioned before due to a large number of features in q_story when compared to our q_remaninder data set size. Shown below is our learning curve with validation accuracy in relation to iteration weight.



Prediction:

Since we are only given three images then we expect that at minimum our model will outperform random prediction; at minimum, it will have a ⅓ == 33.33 percent accuracy. Realistically we should be able to outperform the random guessing accuracy by at least two times; thus we believe that we will achieve ~66 percent testing accuracy. We believe that we will be able to achieve this as we achieved a testing accuracy of ~0.7113… (as shown below), and even possibly performing poorly on future datasets we should still be able to achieve our lower bound realistic accuracy of 66 percent.

```
~/Py/c/d/models python3 FinalModel.py ✔ base 19:18:41
Test Acc : 0.711340206185567
```

FinalModel.py

Limitations, further improvements & conclusion:

Although our final model performed relatively well when making a prediction for the test set. There are still limitations in our model selection and the data.

In the model section of the report, we mentioned that the performance of Logistic regression performs similarly to Naive-Bayes; the difference is negligible. However, implementing logistic regression for model 1 would be a reasonable choice since logistic regression would be more likely to perform a high variance, low bias prediction. This would compensate for model 2 which uses Naive Bayes that will perform a high bias, low variance prediction.

For data, a problem that prevents us from achieving a higher result is that the data size is too small, especially for model 2. We have imposed two strategies to fix the problem; the first is to perform bootstrap resampling to increase our data size, and for the second strategy we used a simple 1D convolution kernel to reduce the number of features. However, it didn't improve the performance of our model. Further improvements to our data set are possible but cannot be achieved within the scope of this course and our technical abilities. (see: BernulliBayesStory(1).py, BernulliBayesStory.py, process_data.py)

Workload Distribution:

Jiaxu Li: Contributed to data preprocessing and kept the working journal. Organizing the workflow of the whole project. Revising the final report based on the code.

Yanke Mao: Working for EDA, Model Exploration, Model Design and Tuning.

Yuran Zhang: Leads the Model Exploration, working on model development and generating the learning curves.

Alex Denisov: Wrote csc311challenge.pdf / report.pdf report and helped Yuran with general coding.